TWO MAJOR ISSUES IN DATA GRID REPLICATION PROCESS

Ahmad Raf'ie Pratama[1]
[1]*Department of Informatics, Faculty of Industrial Technology, Islamic University of Indonesia*
Jl. Kaliurang Km. 14 Yogyakarta 55501
Phone. (0274) 895287 ext. 122, Fax. (0274) 895007ext. 148
E-mail: ahmad.rafie@fti.uii.ac.id

ABSTRACT
This paper discusses Data Grid as one of popular application of grid and cloud computing. In Data Grid, data are replicated among different nodes of the grid to increase the availability and efficiency. There are two main issues regarding this replication process; how to do the replication (i.e. where and when to do replication) and how to synchronise all the replicas under heterogeneous database systems in grids to be always consistent. This paper explains those two major issues as well as some proposed methods and solutions in order to explore the next problems and challenges on this area. By identifying them, it can be useful to make some improvements in order to give the best performance on the replication process, and hence can offer a better experience of Data Grid implementation to the users.

Keyword: Data Grid, replication, replica placement, replica synchronisation

1. INTRODUCTION

1.1 Introduction of Data Grid

Grid is described as a form of distributed computing technology which 'coordinates resources that are not subject to centralized control using standard, open, general-purpose protocols and interfaces to deliver nontrivial qualities of service.' (Foster, What is the Grid? A Three Point Checklist, 2002). In grids, many different people and organisations share their resources to work together in a Virtual Organisations with some defined sharing rules. (Foster, Kesselman, & Tuecke, The Anatomy of the Grid: Enabling Scalable Virtual Organizations, 2001). The resources shared over the grid can be anything, it could be hardware, software, or even the data stored under the system. Grid offer great opportunity to join separated computational resources to work together synchronously and deliver a huge computational power to the users who need it. Currently, grids have been implemented widely in many systems which need large computational resources with fast processing time. Grids environment, including the applications running over the grid is supported by middleware services from a community-based set of libraries called Globus Toolkit (Foster, Kesselman, & Tuecke, The Anatomy of the Grid: Enabling Scalable Virtual Organizations, 2001).

Since it is a form of distributed computing technology, grid is then very suitable environment to deploy a distributed database system. In fact, Data Grid is now one of popular grid application where people can deal with ultra-large-scale of data stored, managed, and processed over many different nodes inside the grid (Chervenak, Foster, Kesselman, Salisbury, & Tuecke, 1999).

This paper explains the replication process in Data Grid, identifies the two major issues on the replication process itself, and discusses some of proposed methods in handling these major issues. The objectives of this research is to explore the next problems and challenges possibilities on this area, which in turns can be useful to make some improvements of current solution alternatives. These improvements are needed to give the best performance on the replication process, so it can offer a better experience of Data Grid implementation to the users.

1.2 Replication Process in Data Grid

Data Grid is a form of distributed database management systems implemented in grid environment. The general architecture of data grid can be seen in Figure 1. In Data Grid, multiple database systems stored in different locations are coordinated and synchronised automatically by Grid Services and keep its transparency to the users as well as to the grid applications.

In order to handle a huge amount of data inside a Data Grid, data replication process as illustrated in Figure 2 is the most common optimisation method used widely on the implementation. It gives some benefits in increasing data availability, reducing access time, reducing bandwidth consumption, and also providing load balancing support (Lamehamedi, Szymanski, Shentu, & Deelman, 2002). In Globus Toolkit, data replication is supported by the Globus Replica Catalogue which job is to keep track of all multiple physical replicas for each logical file inside the grid. It maintains all mapping between logical file names and the physical locations (Stockinger, Samar, Allcock, Foster, Holtman, & Tierney, 2001).

ISSN: 1907-5022

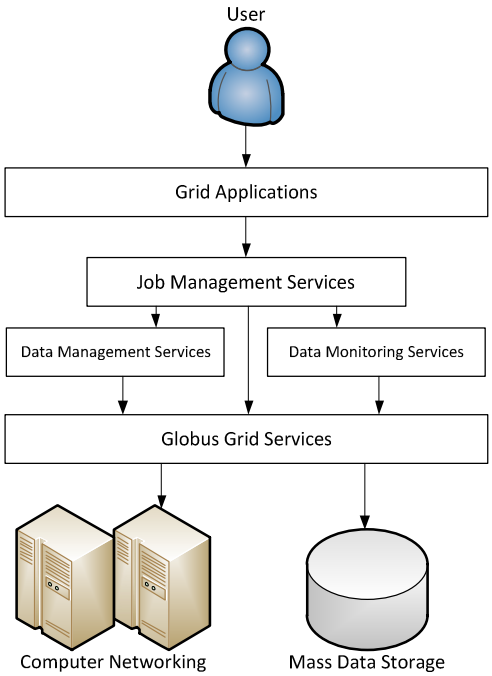

Fig.1. General architecture of data grid.

However, despite the benefit given by replication to Data Grid, the replication process itself still has some issues which can affect the performance or even availability of all services provided by Data Grid. There are two major problems which will be discussed further in this paper. The first one is how to do the replication. In which node replication should be done, and when. Placing the replica in appropriate place and time will improve efficiency by reducing bandwidth consumption and access time for the user making queries. Therefore, the performance of a Data Grid will be much influenced by the replica placement method being used. The second issue to be solved is regarding synchronisation between all replicas within Data Grid.

Furthermore, since the data sets inside Data Grid are usually not supposed to just be read-only data but are updateable data instead, thus synchronisation becomes important to keep all replicas to be always consistent. Furthermore, this issue should also consider that Data Grid usually consists of heterogeneous database systems which cannot talk to each other directly; there should be a mechanism provided to translate between different SQL dialects used by different database systems. These two issues currently can be considered as major issues to be solved regarding the replication process within Data Grid in order to give a better performance to the users (i.e. high availability rate, low latencies, and high efficiency).

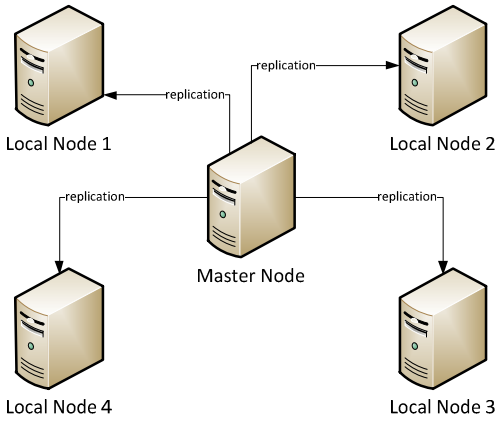

Fig.2. Illustration of replication process

2. REPLICATION METHODS

There are some methods proposed by some researchers to overcome the two major issues of replication process within Data Grid. Some of them will be discussed further on the other sections below.

2.1 Replica Placement Algorithm

Data Replication process needs a Replica Placement Algorithm in deciding where and when to do replication, as illustrated in Figure 3. In general, replica placement algorithms can be divided into two categories; static and dynamic. In static algorithm, the decision on where and when to create replica(s) is made by considering certain factors which are expected may reduce the access time or give the lowest cost, anything which can improve the performance of Data Grid overall. Meanwhile, dynamic algorithm uses more sophisticated considerations depending on some conditions in making the decisions; therefore the decisions may vary for each different time. Dynamic replica placement algorithms can deliver better performance compare to the static ones since it includes more factors and do not stick on one rule only. However, static replica placement algorithms which are much simpler can benefit from fast decision making since there are not many things to be considered.

Lamehamedi et al. (2002) proposed an algorithm which will create replicas at nodes with high number of request. It uses local replica indices containing file information maintained in all nodes. While the list of all replica locations is contained in global replica index maintained in the set root. On the other hand, Naseera and Murthy (2009) used agent-based system model on deciding which site is the best for the placement of a new replica. The agent will consider some resource factors (i.e. Baud-Rate between sites, CPU Rating, CPU Load, Site Storage Capacity, and Local Demand of the replicas at each site) which affect data transmission time between the sites in making the decision.

There are also some dynamic replication strategy have been proposed so far. Lie et al. (2006)

ISSN: 1907-5022

proposed an algorithm called MinDmr Optimizer which addressed to overcome limited replica storage problem in maximising data availability by minimising the data missed rate. Meanwhile, Sashi and Thanamani (2010) proposed dynamic replica creation algorithm which consists of two stages; determining which file to be replicated and then calculating the number of copies to be replicated. This algorithm uses Access Frequency to identify popular files and divide it with average Access Frequency of all other files to determine the number of replicas to be created. Afterwards, this algorithm will place a replica in site with low placement cost which can be calculated from the number of request made for the particular file and the response time.

On the other hand, Abdullah et al (2008) proposed some decentralised replication strategies which are addressed for Peer-to-Peer (P2P) based Data Grid; *Path and Requestor Node Placement Strategy* and *N-hop Neighbour Node Placement Strategy*. The first strategy will create replicas on the requestor node and all nodes along the path between the requestor node and the provider node once a search succeeds. Meanwhile, the second strategy will create replicas on all neighbours of the provider node within N-hop range.

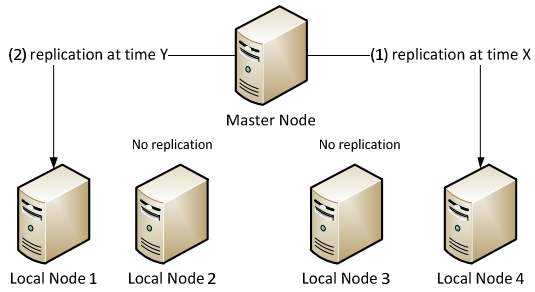

Fig.3. Illustration of replica placement decision

Overall, there are many kinds of replica placement algorithm have been proposed so far. There are static and dynamic algorithms, centralised and decentralised algorithms, and there are also some algorithms developed to work on specific topology of Data Grid, for example P2P topology. Nevertheless, all of them have one similarity; they have one goal of increasing the performance of Data Grid by considering various factors and using different strategies.

2.2 Replica Synchronisation Algorithm

Replication is undoubtedly a useful method of improving the performance of distributed database system, including Data Grid. However, replication introduces another issue when we need to update the database. By having some replicas of a file means that any updates occur to any files should be followed by updating all the replicas as well. Otherwise, there will be inconsistency issue of the service given by the Data Grid. Therefore, replication should always be followed by replica synchronisation service as can be seen on Figure 4.

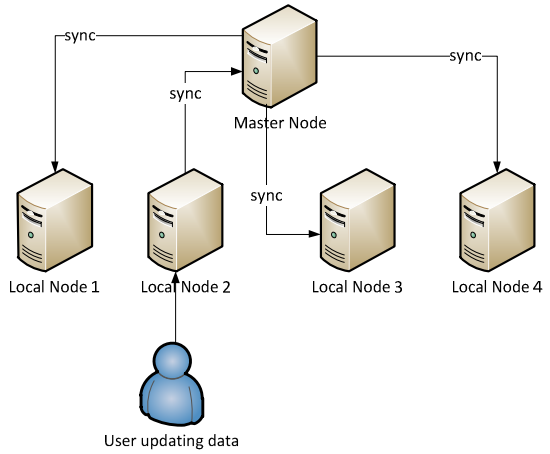

Fig.4. Illustration of replica synchronisation process

There are two basic replica synchronisation strategies in managing distributed database; Strict Synchronisation and Lazy Synchronisation (Ozsu & Valduriez, 1999). Strict means the consistency should be well-maintained; any updates on a single file should be always followed by updating all the replicas immediately, it is also called Synchronous algorithm. Meanwhile, in Lazy or Asynchronous algorithm, any updates happened on any file will be propagated to the replicas after certain periods of time.

Currently, there are some methods have been proposed in replica synchronisation on Data Grid. One of the popular ones is CONStanza (Pucciani, Domenici, Donno, & Stockinger, 2010). This method is built based on optimistic or lazy algorithm. The architecture of CONStanza consists of Global Replica Consistency Service (GRCS) as the core server with its Replica Consistency Catalogue (RCC) in master site and some Local Replica Consistency Services (LRCSs) with their Local Replica Consistency Catalogues (LRCCs) in slave sites. GRCS maintains all the information regarding update processes and dispatches update requests to LRCSs. Meanwhile, LRCSs are charged to maintain the slave databases on their local sites; update requests are received from either GRCS or their DBUpdater in their own local site.

The other model has been proposed is called RUPAGATION (Ciglan & Hluchy, 2007). In this model, synchronisation service is done by Replica Update Propagation (RUP) Service which is broken down into several modules; Update Construction Module (UCM) which will catch any update statements and construct update statement sets which will be propagated later to the other servers, Update Applier Module (UAM) which will do pre-processing of any incoming update sets from other servers and then apply it on its data resource, Update Transfer Module (UTM) which is responsible on

transferring update sets between sites, and Replica Update Manager Module which will act as the manager of all previous modules. On the upper layer of RUP, there is Replica Consistency Service (RCS) as the environment of replica consistency policy implementation from the user.

These two methods use different strategies on handling replica synchronisation on Data Grid. However, both of them focus on delivering consistency of all the replicas in certain degree. In term of heterogeneity support, CONStanza uses a translation module based on a parser included within DBWatcher component, while RUPAGATION use only standard SQL statements when doing synchronisation with help of integration of external SQL dialect translation module.

3. IMPLEMENTATION

In order to overcome the two major issues of replication process within Data Grid, there are several different methods have been proposed by researchers which will be discussed further below.

3.1 Replica Placement Implementation

Different approaches of replica placement implementation may give different decisions on where and when to create replica(s). It depends on what factor to be considered as the most important factor which can benefit users and give them the best experience using the Data Grid. Simulation by Lamehamedi et al. (2002) showed that their algorithm gives better performance when replicas are created closer to the users with larger file size which may offer up to 12% improvement in matter of response time. This simulation was done by comparing three scenarios of data replications on two-tier tree topology; without replication, replication at intermediate nodes, and replication at lower level nodes. Each scenario was processing eight different file sizes ranging from 100MB up to 1GB.

Another simulation by Naseera and Murthy (2009) showed that their algorithm suggested Site1 of 10 Sites used in their simulation as the best candidate of replica placement site based on lowest aggregated execution time test compared to the other sites. However, when it comes to data availability test, the algorithm suggested Site 7 which offers lowest aggregated data transmission time compared to the other sites, even to Site 1. Therefore, they used weighting factor in making the decision which can be selected by the users depends on their need. In that case, when users concern more in execution time, they will get the replication done on Site 1, while Site 7 will be the place of replication if they concern more in file transmission through the network.

In short words, replication was proven to give better performance to the Data Grid. However, users should know their requirement and pick the most suitable approach with also some adjustment to give them more benefit since inappropriate approach may result in small improvement which is not significant enough to them.

3.2 Replica Synchronisation Implementation

Both CONStanza and RUPAGATION used different methods on measuring their performance on the implementation step; CONStanza used measuring of the time (milliseconds) needed to perform synchronisation of 1, 10, 100, and 1000 updates within five different sites, while RUPAGATION used comparison of the time needed for updating 1000 files in both non-replicated system and replicated system which includes synchronisation to keep all replicas to be consistent. Furthermore, while CONStanza have been implemented in several Data Grid systems so far, RUPAGATION still used a prototype model on the implementation step.

In addition, based on a simulation using CONStanza, four distributed MySQL slave databases were successfully synchronised with an Oracle master database within no more than 12 seconds for 1000 of updates. This synchronisation time is majorly constituted from the update file creation step (Pucciani, Domenici, Donno, & Stockinger, 2010). Meanwhile, in a simulation of updating 1000 files using RUPAGATION with two replica sources, the synchronisation took up to 14.5 times more time compare to the time of updating 1000 non-replicated files (Ciglan & Hluchy, 2007). Regarding heterogeneity support, RUPAGATION was proven to support more SQL dialects including XML databases, it has been tested by using three kinds of database systems; MySQL, PostgreSQL, and eXist. Meanwhile, CONStanza was only implemented to work with two types of relational databases systems; Oracle and MySQL.

Nevertheless, both methods have similarity on the need of some improvement, primarily in handling larger sets of database, having more than 1000 updates at a single time. This improvement is urgent since in real applications, Data Grid will have ultra large size of database with millions of records and replicas, thus give high possibility of having big number of updates within shot timeframe.

4. CONCLUSION

4.1 Identified Major Issues

Replication is very popular optimisation method in Data Grid. It can increase the availability of the data needed by users. However, replication also introduces two major issues which can reduce the performance of the Data Grid instead of increases it. The first issue is regarding how to do the replication, and the second one is on synchronising all the replicas to keep them in consistency.

Replica placement algorithm is needed in deciding where and when to create replicas. There are some algorithms using various methods being used so far. They include some different parameters to make some calculations in making the decisions. It can be a centralised method or a decentralised method. All of them have the same goal; creating replica on appropriate place(s) and time so the replication will be done effectively, increase the availability rate, reducing access time, and therefore improve the performance of the Data Grid.

Replication also introduces another issue when there is a need of data updating. Having more than one replica means that any update occurs to a single file should be followed by updating all other replicas as well. Replica synchronisation mechanism is needed to keep all these replicas in consistency. Synchronisation in Data Grid should consider some factors, e.g. the large size and/or number of databases, and also the heterogeneity of database systems. CONStanza and RUPAGATION are two methods of replication synchronisation developed for Data Grid. Both of them use different strategies and have different capabilities, CONStanza offers better performance while RUPAGATION has more feature and wider support in term of heterogeneity.

4.2 Future Research Opportunity

Although there are many methods have been proposed to handle those two major issues on Data Grid, there are still many chances to make some improvements on these methods. In term of replica placement, one of the chances is on how to get a better performance of Data Grid by inventing new method of replication which can increase availability rate at the lowest cost. On the other hand, CONStanza and RUPAGATION have succeeded in providing a good service in term of replica synchronisation. However, these two methods still need to be improved further, particularly in term of scalability; how to deal with much larger database query without suffering much from low performance & slower speed of access time.

REFERENCES

Abdullah, A., Othman, M., Ibrahim, H., Sulaiman, M., & Othman, A. (2008). Decentralized replication strategies for P2P based Scientific Data Grid. *Information Technology, 2008. ITSim 2008. International Symposium on* (pp. 1-8). Kuala Lumpur: IEEE Computer Society.

Chervenak, A., Foster, I., Kesselman, C., Salisbury, C., & Tuecke, S. (1999). The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scienti. *Journal of Network and Computer Applications,* 187-200.

Ciglan, M., & Hluchy, L. (2007). Content Synchronization in Replicated Grid Database Resources. *Third International IEEE Conference on Signal-Image Technologies and Internet-Based System* (pp. 379-386). Shanghai: IEEE Computer Society.

Dullmann, D., Hoschek, W., Jaen-Martinez, J., Segal, B., Samar, A., Stockinger, H., et al. (2001). Models for Replica Synchronisation and Consistency in a Data Grid. *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing* (pp. 67-75). San Francisco: IEEE Computer Society.

Foster, I. (2002, July). What is the Grid? A Three Point Checklist.

Foster, I., Kesselman, C., & Tuecke, S. (2001). The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications, 15(3)*, 200-222.

Lamehamedi, H., Szymanski, B., Shentu, Z., & Deelman, E. (2002). Data Replication Strategies in *Grid Environments. Algorithms and Architectures for Parallel Processing, 2002. Proceedings. Fifth International Conference on* (pp. 378-383). Beijing: IEEE Computer Society.

Lei, M., Vrbsky, S. V., & Hong, X. (2006). A Dynamic Data Grid Replication Strategy to Minimize the Data Missed. *Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on* (pp. 1-10). San Jose: IEEE Computer Society.

Naseera, S., & Murthy, K. (2009). Agent Based Replica Placement in a Data Grid Environement. *Computational Intelligence, Communication Systems and Networks, 2009. CICSYN '09. First International Conference on* (pp. 426-430). Indore: IEEE Computer Society.

Ozsu, M. T., & Valduriez, P. (1999). *Principles of Distributed Database Systems.* New Jersey: Prentice-Hall, Inc.

Pucciani, G., Domenici, A., Donno, F., & Stockinger, H. (2010). A Performance Study on the Synchronisation of Heterogeneous Grid Database using CONStanza. *Future Generation Computer System,* 820-834.

Sashi, K., & Thanamani, A. (2010). A New Replica Creation and Placement Algorithm for Data Grid Environment. *Data Storage and Data Engineering (DSDE), 2010 International Conference on* (pp. 265-269). Bangalore: IEEE Computer Society.

Stockinger, H., Samar, A., Allcock, B., Foster, I., Holtman, K., & Tierney, B. (2001). File and Object Replication in Data Grids. *High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium on* (pp. 76-86). San Fransisco: IEEE Computer Society.